We know how to use modulo division with positive integers, but what about the negative ones? What number do we consider as the remainder if we divide −42 by 10 or 42 by −10? Is the concept the same across all programming languages? That's not an idle question because a lot of practical algorithms (for example, those used in data encryption) contain modulo division. So, a lack of knowledge may lead to dangerous mistakes in development.

**Recap**

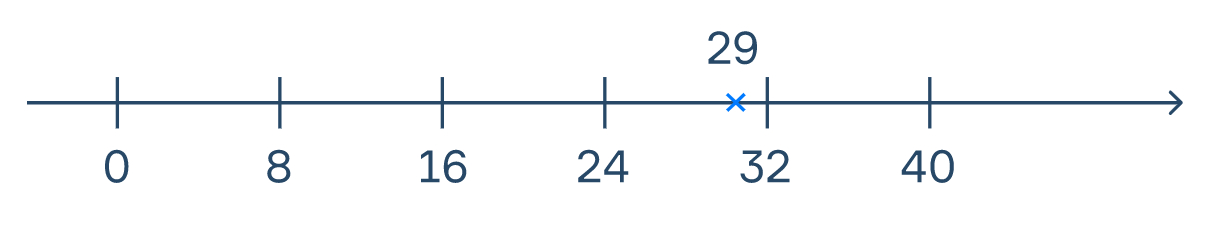
Suppose we want to divide 29 by 8. From what we already know about modulo division with positive integers, we can work out the remainder of this division:

29 / 8=3.62529 / 8=3.625

After rounding down the result, we obtain our integer quotient: 3. Then, the remainder is:

dividend=(divisor⋅quotient)+remainder⇒remainder=dividend−(divisor⋅quotient)=29−(8⋅3)=29−24=5dividend=(divisor⋅quotient)+remainder⇒remainder​=dividend−(divisor⋅quotient)=29−(8⋅3)=29−24=5​

A more visual way to look at it is to place all positive multiples of 8 (that is 8, 16, 24, 32, 40...) on the number line like so:

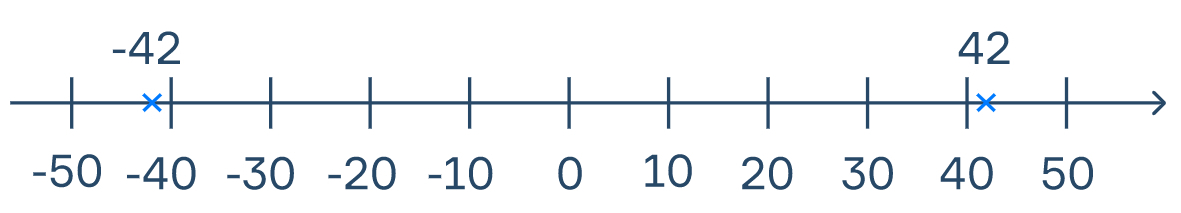


We see that 29 lies between 24 and 32. We can see that the integer part of the result is equal to 24, as 24=8⋅324=8⋅3. The remainder corresponds to the distance between our dividend and this number: 29−24=529−24=5.

Can we generalize this to negative numbers? Yes, but this will result in different definitions of modulo division! Let's take a look at each of them and try to answer our first question of dividing−42 by 10 (and 42 by −10, while we're at it).

**Euclidean division**

We'll begin with our number line but now we'll write multiples of 10. Also, this time we will consider all multiples, not just the positive ones. Let's take a look:



We can see that −42 lies between -50 and −40. Since −50=(−5)⋅10−50=(−5)⋅10, the integer quotient here is -5.

We can also see that 42 lies between 40 and 50. Since 40=(−4)⋅(−10)40=(−4)⋅(−10), the integer quotient here is -4.

This approach is called the **Euclidean division**, or **E-division** for short. We can describe the algorithm for dividing a*a* by b*b* as follows:

1. Find the greatest multiple of b*b* which does not exceed a*a* .
2. Suppose that multiple is q⋅b*q*⋅*b*. Then q*q* is the quotient.
3. The remainder r*r* will be equal to a−q⋅b*a*−*q*⋅*b*.

In this way, the dividend can be presented as

a=q⋅b+r;r≥0*a*=*q*⋅*b*+*r*;*r*≥0

Note that the remainder will always be non-negative when using E-division

Thus, we have

a.(−42)/10 ⇒−42=(−5)⋅(10)+r ⇒ r=50−42=8 b.42/(−10) ⇒42=(−4)⋅(−10)+r ⇒ r=42−40=2a.(−42)/10 ⇒−42=(−5)⋅(10)+*r* ⇒ *r*=50−42=8 b.42/(−10) ⇒42=(−4)⋅(−10)+*r* ⇒ *r*=42−40=2

Usually, when mathematicians talk about modulo division, they mean Euclidean division. It has some nice properties: for example, if you take any two numbers whose difference is a multiple of b*b*, they'll have the same remainder when you divide each of them by b*b*.

**Floored and Truncated**

E-division is also interesting because it does not use real numbers as if there were only integers in our world. However, what if we worked backwards from the floating-point division?

We know that

(−42) / 10=42 / (−10)=−4.2(−42) / 10=42 / (−10)=−4.2

So now, in order to determine the remainder, we first need to define the integer quotient from this result. We can do that in different ways, but let's explore the two most common:

- **Truncated division**, or **T-division**: as its name suggests, the quotient is obtained by truncating the result. Here, we take only the integer part of the result, essentially "erasing" all digits after the decimal point without performing any rounding operation. In our example, if we take the integer part only, we can see that the quotient is −4 for both cases.

- **Floored division**, or **F-division**: here, the "floor" corresponds to the closest integer less than or equal to the result. Thus, the integer quotient is obtained by rounding the result to its corresponding floor. In our example, the closest integer less than -4.2 is -5, so in both cases, the quotient is -5.

Different types of division give us different results. However, note how now we get the same integer quotient for both (−42) / 10(−42) / 10 and 42 / (−10)42 / (−10), in contrast to Euclidean division.

So, what about the remainders? Following the dividend formula from before, we know that

a=q⋅b+r⇒r=a−q⋅b*a*=*q*⋅*b*+*r*⇒*r*=*a*−*q*⋅*b*

Then, for the truncated division we have

a.(−42)/10 ⇒−42=(−4)⋅(10)+r ⇒ r=40−42=−2b.42/(−10) ⇒42=(−4)⋅(−10)+r ⇒ r=42−40=2​a.(−42)/10 ⇒−42=(−4)⋅(10)+*r* ⇒ *r*=40−42=−2b.42/(−10) ⇒42=(−4)⋅(−10)+*r* ⇒ *r*=42−40=2​

And, for the floored division we have

a.(−42)/10 ⇒−42=(−5)⋅(10)+r ⇒ r=50−42=8b.42/(−10) ⇒42=(−5)⋅(−10)+r ⇒ r=42−50=−8​a.(−42)/10 ⇒−42=(−5)⋅(10)+*r* ⇒ *r*=50−42=8b.42/(−10) ⇒42=(−5)⋅(−10)+*r* ⇒ *r*=42−50=−8​

**They are so different...**

You have probably already noticed how both the integer quotient and the remainder values depend on how we define the division operation. We can compare the three types with more examples:

| a.11 / (−4)a.11 / (−4) | | | |
| --- | --- | --- | --- |
| **Division** | **Quotient (Integer division)** | **Remainder (mod, %)** | **Complete expression a = q · b + r** |
| **Euclidean** | -2 | 3 | 11=(−2)⋅(−4)+311=(−2)⋅(−4)+3 |
| **Truncated** | -2 | 3 | 11=(−2)⋅(−4)+311=(−2)⋅(−4)+3 |
| **Floored** | -3 | -1 | 11=(−3)⋅(−4)−111=(−3)⋅(−4)−1 |

| b.(−11) / 4b.(−11) / 4 | | | |
| --- | --- | --- | --- |
| **Division** | **Quotient (Integer division)** | **Remainder (mod, %)** | **Complete expression a = q · b + r** |
| **Euclidean** | -3 | 1 | −11=(−3)⋅(4)+1−11=(−3)⋅(4)+1 |
| **Truncated** | -2 | -3 | −11=(−2)⋅(4)−3−11=(−2)⋅(4)−3 |
| **Floored** | -3 | 1 | −11=(−3)⋅(4)+1−11=(−3)⋅(4)+1 |

| c.(−11) / (−4)c.(−11) / (−4) | | | |
| --- | --- | --- | --- |
| **Division** | **Quotient (Integer division)** | **Remainder (mod, %)** | **Complete expression a = q · b + r** |
| **Euclidean** | 3 | 1 | −11=(3)⋅(−4)+1−11=(3)⋅(−4)+1 |
| **Truncated** | 2 | -3 | −11=(2)⋅(−4)−3−11=(2)⋅(−4)−3 |
| **Floored** | 2 | -3 | −11=(2)⋅(−4)−3−11=(2)⋅(−4)−3 |

| d.11 / 4d.11 / 4 | | | |
| --- | --- | --- | --- |
| **Division** | **Quotient (Integer division)** | **Remainder (mod, %)** | **Complete expression a = q · b + r** |
| **Euclidean** | 2 | 3 | 11=(2)⋅(4)+311=(2)⋅(4)+3 |
| **Truncated** | 2 | 3 | 11=(2)⋅(4)+311=(2)⋅(4)+3 |
| **Floored** | 2 | 3 | 11=(2)⋅(4)+311=(2)⋅(4)+3 |

The results are the same for all three types of division when both the dividend and the divisor are positive.

Different programming languages use different definitions for modulo division. It is essential to know this if you are switching between them from time to time. Here are some of the most popular ones with their corresponding division type:

|  |  |
| --- | --- |
| **Language** | **Division** |
| Python | Floored |
| Ruby | Floored |
| Java | Truncated |
| Kotlin | Truncated |
| C++ | Truncated |

Why don't we see it for ourselves?

Run the following code in Python:

print(431 % -6)

you will see that −1 will be printed to the console, which is the result of Floored division. Python uses the F-division definition in its implementation of the modulo operator.

Now, let's run the following code in Java:

public class Main {

public static void main(String[] args) {

System.out.println(431 % -6);

}

}

here 5 will be printed instead. This is because Java uses the Truncated division definition for its implementation of the modulo operator.

For a comprehensive list of the definition used by all programming languages